UNLIMITED

RSRE
MEMORANDUM No. 4293

ROYAL SIGNALS & RADAR ESTABLISHMENT

THE INTEGRATION OF STRUCTURED AND FORMAL METHODS

Author: G P Randell

PROCUREMENT EXECUTIVE,
MINISTRY OF DEFENCE,
R S R E MALVERN,
WORCS.

RSRE MEMORANDUM No. 4293

CONDITIONS OF RELEASE

0051799 BR-111533

********** U

********** Y

Reports quoted are not necessarily available to members of the public or to commercial organisations.

ROYAL SIGNALS AND RADAR ESTABLISHMENT

Memorandum 4293

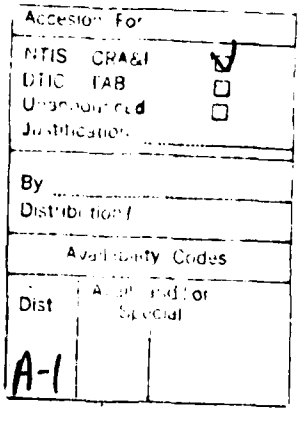
Title: The Integration of Structured and Formal Methods

Author: G P Randell

Date: June 1989

Summary

A growing emphasis is being put on a formal, in the sense of mathematically rigorous, approach to the development of software based systems. However, current formal "methods" tend to be just notations. This memorandum looks at several structured methods and considers the possibilities each offers for integration with a formal notation to produce a usable formal method combining the best of both formal and structured approaches. Proposals for further work are discussed.



CONTENTS

1. Introduction

Requirements specifications for software-based systems must accurately reflect the users' real requirements, and should be complete, consistent and unambiguous. In order to achieve this, good systems engineering methods need to be developed and applied during the process of producing a specification. A systems engineering approach is the use of engineering principles to analyse, plan, create and support a system. A systematic and structured approach needs to be followed throughout the whole life-cycle of a system in order to reduce such problems as timescale overrun and the exceeding of budgets which may affect large projects.

Both structured and formal methods exist for use in the production of requirements specifications. The commonly used structured methods are SSADM, JSD, Yourdon and CORE. These methods comprise various diagrammatic notations for the expression of requirements, and guidelines on the method of using these notations. Some structured methods take a much wider view than simply producing a specification, and seek to control the whole process of developing a specification, including feasibility studies and quality assurance reviewing.

The formal methods at the current stage of their development tend to be notations and have very little method associated with them, at least as far as the production of a specification is concerned. I will use the term "formal notation" in this report. Commonly used formal notations are Z, VDM and OBJ. Structured methods and formal notations can be considered to be at opposite ends of a spectrum. Formal notations are mathematically precise but require understanding of mathematics to use them, whereas structured methods, at present, lack rigour but are found by many software engineers to be easier to use and, in the methodology, deal with many commonly encountered engineering decisions.

A growing emphasis is being put on a formal, in the sense of mathematically rigorous, approach, especially in the areas of secure and safety-critical systems. This is because a formal specification is an unambigous statement which enables propositions about the system to be proved mathematically. It is therefore of interest to study means of integrating formal notations and structured methods, to produce a formal method which is easier to use and which fits in with engineering practice, incorporating the best of both approaches.

To this aim, this study looks at several existing structured methods and considers the possibilities each offers for such an integration. The characteristics looked for in these methods are discussed in section 3 below.

The remainder of the report is structured as follows. Section 2 gives an introduction to a systems engineering approach to set the scene for the study. Section 4 presents the assessment of the selected methods, and sections 5 and 6 discuss the results and future work. A description of each method appears in an accompanying memo [Randell].

2. Systems Engineering

The systems engineering approach presented in this section reflects the opinions of representatives from the MoD Research Establishments [C3ISSC]. The starting point of the systems engineering approach is a framework of activities. This framework identifies the activities which may need to be undertaken during the development of a system, their resultant products, and the relationship between the activities.

The activity of interest is that of the specification of the requirements, which is part of the **Requirements Definition** stage of the **System Acquisition** activity. The system acquisition activity has seven stages. The first

of these is project management, where the planning, estimating, initiation, monitoring and controlling of the other activities is carried out. The second stage is the requirements definition stage, and the objective of this stage is to determine and record what is required of the system. This includes both functional properties and system constraints (e.g. cost, timescale, performance, security). These latter are also termed "non-functional requirements". The major output is the system requirements specification, which should be a clear, consistent, complete, precise and unambiguous statement of the requirement. The fundamental processes of the requirements definition activity are:

Requirements Acquisition: the acquisition, or capture, of information from the user (which may be more than one person or organisation).

Requirements Expression: the expression of this information in a notation which elucidates implications, correlates different aspects and facilitates detailed design.

Requirements Analysis: the analysis of the expressed requirements for internal consistency, completeness and precision. This may lead to further consultations with the user if conflicting requirements are discovered.

Production of the Requirements Specification: the consolidation of all relevant information obtained from the above three processes into a well structured format to serve as the definitive statement of what the system is required to do and the basis for architectural design.

The requirements specification is input to the third stage of system acquisition, namely the architectural (or top-level) system design stage. In this stage the plans of the system to be built to meet the specification are devised. In the fourth stage, unit design, detailed design of the separate units specified in the previous stage is carried out.

The fifth and sixth stages are those of unit construction and system integration where the system is built and tested. The final stage is that of acceptance. The relationship of all the activities of the system engineering approach are shown in Figure 1.

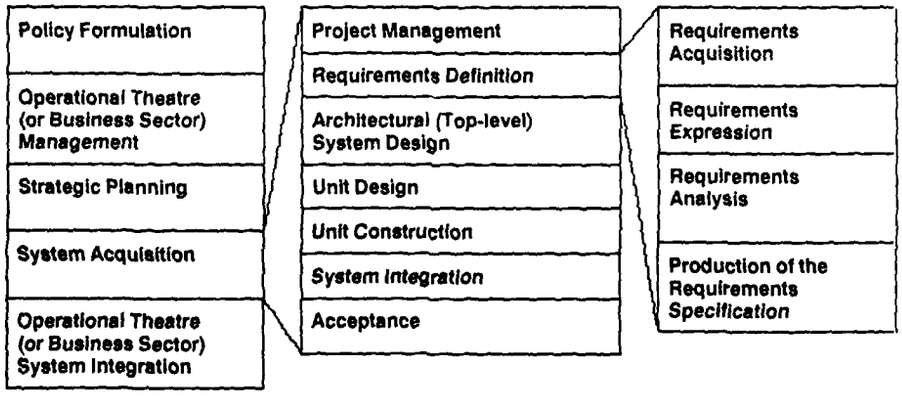


Figure 1 Systems Engineering Activities

To aid thinking on the topic of systems engineering, a layered model (Figure 2) has been proposed [Bates]. The model is not complete, but is a first attempt.

In this model the functions undertaken in level *n* make use of the functions supported in the lower layer *(n-1)* and provide support to the higher layer *(n + 1)*. The first (lowest) layer contains the common computer based

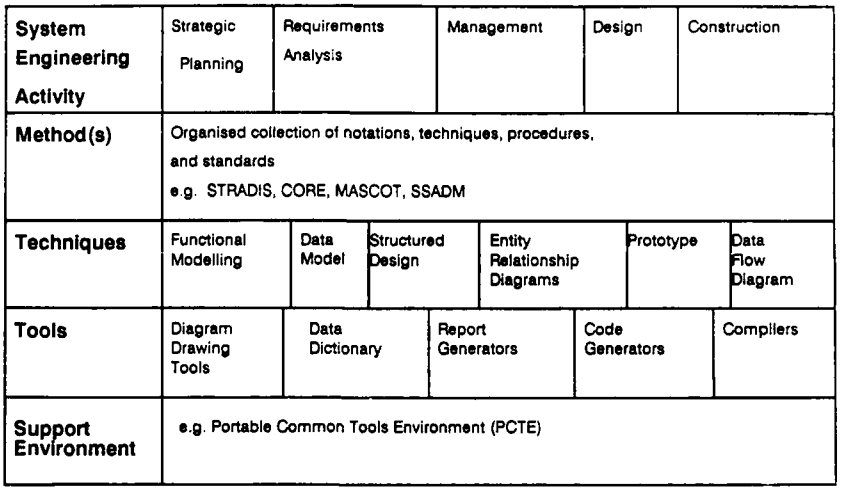


Figure 2 Systems Engineering Layered Model

support environment. Sitting on top of this are the tools (that is, software packages) for diagram drawing, consistency checking etc. Such tools use the services of the support environment and provide support to the third layer, the techniques such as data modelling, prototyping etc. A single technique may require the use of several tools. A single tool may support more than one technique.

The fourth layer contains the methods (that is, organised collections of notations, techniques, procedures and standards). Many of these use similar techniques, but perhaps in a different order or to different effect. Most methods at present support only a single (or small number) of the system engineering activities (layer 5). Consequently, a compatible set of methods is required to support all of these activities undertaken during the development process.

It should be noted that the formal notation Z would appear in the techniques layer of this model, not the method layer. Z is supported by editors and type-checkers, and would also benefit from the support of theorem provers, but is a notation, not a method. At this stage in its development, the Z "method" is no more than the practice of the Z exponents, but a more systematic approach is supported by VDM [Jones]. A layered model appropriate to a formal approach to the development process is shown in Figure 3.

Good methods and tools should help to reduce costs and to produce systems with fewer undiscovered faults and which are well structured and documented. Such systems are easier to maintain and to modify to meet changing requirements or to upgrade. The use of such methods also supports the construction of more complex systems.

These aims are addressed in the structured methods by demanding of the practitioner a disciplined approach to each of the areas of: accurate realisation of the defined system in its implemented form; recording of information; completeness checking of investigative, analysis, design and construction work; presentation of the requirement (as currently understood), and presentation of the proposed solutions; cross-checking of techniques and results.

The advantage of expressing a requirements specification in a formal notation is that the specification is an unambiguous statement and enables the properties of the system to be made visible as propositions to be proved. The notation improves communication between team members working on the same project as it is unambiguous and concise. The writing of such a specification forces clarity of thought on the implications of requirements, which are written as mathematical statements. With the potential of a refinement process to translate a specification into design and code, there is also the possibility of being able to prove mathematically that a design meets its specification.

System Engineering Activity	Requirements Expression	Requirements Analysis	Architectural Design	Unit Design
Method	Organised collection of notations, procedures, and standards (nearest approach is VDM)			
Techniques	Functional Modelling	Algebraic Techniques	Animation	Refinement
Tools	Editors	Analysis Tools	Theorem Provers	Syntax Checkers	Typecheckers
Support Environment	e.g. Portable Common Tools Environment (PCTE)			

Figure 3 Systems Engineering with a Formal Method

However, the use of formal notations does have some problems. As the requirements expression process should express requirements in a notation which makes them apparent, and elucidates implications, the specification must be understandable to the user. It is possible that a mathematical notation may not be easily enough understood by those without a mathematical background. This makes the process of validation, that is, checking that the specification describes the required system accurately, more difficult.

There is also a problem with the production of a formal specification. As was mentioned earlier, formal methods are really just notations, and offer no help as to the writing of the specification. If such notations are to become widely used, then there must be some support for the activity of producing the requirements specification. There must also be support for checking the consistency and completeness of the specification.

This has set the scene. The interest is in studying existing structured methods to discover which activities of the development process they support, and which techniques they use. The aim is then to consider which offer the most potential to combine in some fashion with a formal notation to provide a useful, workable method combining the best of both structured and formal approaches.

3. The Methods and the Assessment Criteria

The methods chosen for study are:

a) Structured Systems Analysis and Design Method (SSADM)
b) Jackson System Development (JSD)
c) Yourdon
d) COntrolled Requirements Expression (CORE)
e) Soft Systems Approach.

These methods have been chosen because they are in common usage, both within industry and the MoD. Together they are taken to be representative of the common techniques and ideas used by structured methods in general.

The formal notations considered are Z and Communicating Sequential Processes (CSP). These two notations have been chosen as they represent two approaches - Z uses a state-based approach, whereas CSP looks at

processes and their behaviour. CSP is therefore more suitable for specifying concurrent systems. There are many more formal notations in use, among them the Vienna Development Method (VDM), Calculus of Communicating Systems (CCS), and OBJ. However, for the purpose of this study, Z and CSP are being looked upon as representative formal notations.

A more complete study may consider other formal notations. The interested reader is referred to [Strutt] for a list of formal notations currently in use.

There is more than one approach to combining a formal notation with a structured method. Three possibilities are discussed.

Describe the structured method using the formal notation.

The purpose of this is to identify inconsistencies and ambiguities in the method. This would obviously improve the structured method. However, it would require such close interaction with those involved in the designing of the method that it is best carried out by those closer to the actual method.

Use the diagrammatic techniques of the structured method to illustrate the formal specification.

A problem with a formal specification in a notation such as Z is that it may not be readily understandable to those without a background in mathematics. The advantage of the structured methods is that they rely heavily on diagrammatic techniques which are more generally understandable. The idea would be to use some form of diagramming to illustrate a formal specification and make it more understandable.

Generate a formal specification from the diagrams.

This complements the second approach. The idea would be to take some of the diagrams produced by the structured method, and generate an outline formal specification from them. The details of the specification could then be filled in as the next step of the new method, thus giving traceability to the diagrams.

The first of these approaches is concerned with the rules of the method, rather than the individual techniques, in contrast to the second and third approaches. These second two are more concerned with the production and understanding of a formal specification, and are therefore of the most interest.

The production of a specification, be it formal or otherwise, is part of the requirements definition activity. Therefore, any method that does not deal with this area is not relevant to this study.

With these points in mind, the questions which need to be answered for each of the methods are:

> Does the method support the Requirements Definition Activity?
>
> What are the main techniques used by the method in support of this activity?
>
> Is there any existing basis in mathematics for any of the techniques? If not, is there a perceived potential for formalisation?
>
> Have there been attempts to use the method in conjunction with a formal notation?

The aim is to discover which techniques from the methods are the most promising, and then to identify what further work will be needed to meet the goal of a real "formal method".

4. The Assessments of the Methods

In the following sections, the questions posed in section 3 above are answered for each of the selected structured methods.

4.1 Structured Systems Analysis and Design Method (SSADM)

Does the method support the Requirements Definition Activity?

SSADM claims to support the system development lifecycle from the analysis of the current system through to the physical design of the new system. By the end of stage 3 of the method a specification of the chosen system is produced. This corresponds to the production of the requirements specification phase of the requirements definition activity.

The diagramming techniques used in the earlier stages of SSADM offer a means of expressing requirements, however there is little guidance for the process of requirements acquisition. There is also limited support for requirements analysis. The requirements are analysed to the extent that if adding a requirement forces a diagram to disobey the rules of syntax for that type of diagram, then there is an inconsistency which needs to be resolved.

What are the main techniques used by the method in support of this activity?

The techniques used by SSADM to express requirements are diagrammatic, and are:

Logical data structuring, also known as entity modelling, in which a diagram is produced showing the entities of interest and their relationships. This is further documented by a set of entity description forms detailing the data contents of the entities.

Data flow diagrams, which represent the information flows within a system, and show how information enters and leaves a system, what changes the information, and where the information is stored. These diagrams are an important technique as a means of boundary definition. They denote the major functional areas of the system, and therefore the programs or program suites required.

Entity life histories, which are models of how the system's data is changed over time by events acting on entities. For each entity the sequence, selection and iteration of events affecting it are shown using a notation derived from Jackson.

Logical dialogue descriptions, which specify the requirements for human-computer dialogues. These are designed to be used when prototyping facilities (which are more effective) are not available.

The three major techniques of entity modelling, data flow diagrams and entity life histories are used to cross-check one another for consistency and completeness.

Is there any existing basis in mathematics for any of the techniques? If not, is there a perceived potential for formalisation?

None of the four techniques listed above have a basis in mathematics. However, there is potential for formalisation.

There is an automatic translation between a logical data structure and Z. This is because a logical data structure simply represents relationships between entities, and relationships are directly supported by Z, which is based on set theory.

There is also the possibility of translating a data flow diagram into an outline Z specification. To illustrate this

idea, consider the small data flow diagram shown in Figure 4. This diagram shows the process of registering a new customer at a bank. The manager adds the new name to the existing store of customer names.

The Z specification corresponding to this diagram is as follows.

A set of customers is introduced to represent all the possible customers.

[Customer]

The data store, which contains a set of customers which will be called the customer list, is represented by a schema called data_store. A choice has been made to keep the list of customers as a set rather than a sequence or list. This choice has been made not from the data flow diagram, but because a set is the most general structure which may be refined later to a sequence if required. The purpose of the initial Z specification is to be as abstract as possible and avoid making design decisions.

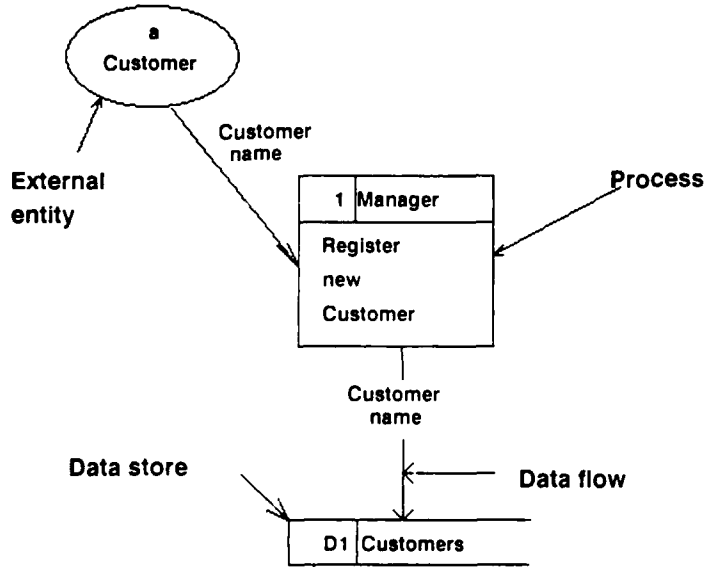


Figure 4 SSADM Data Flow Diagram

data_store
customer_list : ℙ Customer

The operation of registering a new customer takes a customer as input, and updates the data store by adding the new customer to it.

Register_new_customer
new_customer? : Customer
Δ data_store

data_store' = data_store ∪ {new_customer?}

The fact that the operation of registering a new customer is carried out by the manager is lost in this Z specification. However, it could easily be added to the name of the operation if it was thought necessary.

The advantage of the Z specification is that it is possible to add constraints formally as additional predicates in the schemas. For example, a particular bank may decide it can only deal with 49 customers at any one time. The constraint that the data store does not exceed this is the added as a predicate in the schema data_store. In this manner the outline specification generated from the data flow diagram may be refined by adding more information as required.

The Jackson-like notation for the entity life histories may be described using CSP, and this is discussed in section 4.2 below. This would allow a diagram to be translated into its equivalent CSP description, and vice versa.

Logical dialogue descriptions are intended to help with the specification of the human computer interface (HCI). They are a form of data flow diagram, and thus may be able to be treated in a similar manner to the data flow diagrams described above. There are more aspects to the HCI such as timing considerations and other performance requirements which are not dealt with by using logical dialogue descriptions. These would need to be added to the specification, but cannot be done using Z. A form of temporal logic may be more appropriate.

Have there been attempts to use the method in conjunction with a formal notation?

The Structured and Formal Methods Research Project at Leeds Polytechnic is concerned with the use of formal and structured methods in the development of information systems [Bryant]. The underlying beliefs of the project are that: the benefit to be derived from the use of formal methods in the development of information systems is substantial, particularly in the area of specification; the current widespread use of structured methods is a step forward, but these methods still lack the precision to reason about a proposed system at any time in its life; the migration from structured to formal methods appears to be only taking place in a small sector of software product development; and that a potential path of migration is to merge structured and formal methods, where the strengths of each are apparent and thus provide an entry point for current analysts.

The project is concentrating on SSADM and Z. There are three areas of research :

a) Attaching Z as a notation to SSADM at strategic points, notably the Logical Data Structure and the output from the Entity Life History models.

b) Revising SSADM to fully incorporate Z within it.

c) Examining the theoretical underpinnings of SSADM in set-theoretic terms.

4.2 Jackson System Development (JSD)

Does the method support the Requirements Definition Activity?

JSD claims to address most of the software lifecycle. The philosophy of the method's designers is that any statement of requirements probably contains ambiguities and omissions and that the requirements are likely to change. It is therefore proposed that it is better to build a model of the real world and proceed directly to design. This model is in fact the requirements specification, as it describes what the system is required to do.

JSD does not offer any support for the process of requirements acquisition. It offers a diagrammatic notation for expressing the model and for analysing it using the rules of syntax of the diagrams.

What are the main techniques used by the method in support of this activity?

The major technique of JSD is the structure diagram for describing entities and the actions or events which affect them. A process structure is drawn for each entity which describes its life history. This diagramming technique is also used in SSADM.

The other important technique is that of elaborating the process model into a specification by means of a network structure diagram. This brings all the processes together, along with processes for input, output and interactions within the system.

There are additional constraints such as timing which cannot be expressed using the notations of JSD, as they lack sufficient power. Such constraints are added informally.

Is there any existing basis in mathematics for any of the techniques? If not, is there a perceived potential for formalisation?

In his book [Jackson], Michael Jackson acknowledges the influence of work on communicating sequential processes. JSD and CSP are similar, although the former relies more on diagrams while the latter is much more theoretical. While there is no explicit mathematical basis for JSD, it may be that CSP could be a theoretical basis for the concepts and techniques of JSD.

Have there been attempts to use the method in conjunction with a formal notation?

A technical monograph has been published by Sridhar and Hoare [Sridhar] of the Programming Research Group at Oxford University entitled "JSD Expressed in CSP". This studies the relationship between JSD and CSP and suggests that CSP provides a promising theoretical basis for the practical concepts and methods used in JSD. It is hoped that this may suggest fruitful directions for research in the practical application of CSP, and that it may lead to a better appreciation of JSD and perhaps suggest further improvements.

The monograph suggests that there may be a case to introduce some of the more useful CSP operators as new types of boxes in structure diagrams, so that larger systems including parallelism can be described pictorially. The use of such diagrams may be more appealing to those who are unhappy with mathematical notations.

It is claimed that the rich set of laws governing the CSP operators provides a tool to formally derive efficient implementable versions of the specifications.

Three areas of further work suggested are:

a) To formalise the state vector communication of JSD in CSP.

b) To develop techniques to avoid deadlock and establish its absence.

c) To study further the extension of the suggested algebraic transformation approach for deriving efficient implementations of distributed programs to more complicated examples involving arrays of processes. Some form of mechanical aid may be needed.

As a simple example of how a JSD structure diagram may be written in the CSP notation, consider the diagram in Figure 5. This describes part of a banking system, with entity customer and actions invest, pay-in, withdraw and terminate. It represents a customer first opening an account with the action invest. Subsequently, the customer may deposit (pay-in) or withdraw money as many times as is required (overdrafts are permitted) until the account is closed (terminated).

The equivalent CSP definition of process Customer is:

```
Customer = invest -> Customer_body

Customer_body = ( payin -> Customer_body
                | withdraw -> Customer_body
                | terminate -> STOP )
```

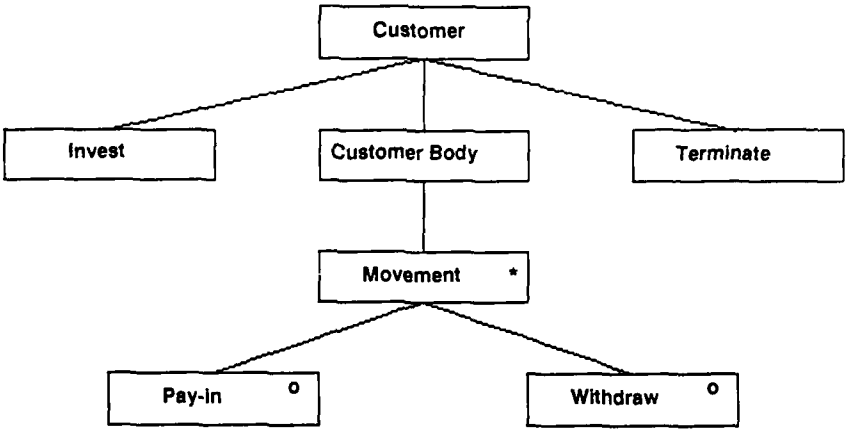


Figure 5 - JSD Structrure Diagram for a Customer

4.3 Yourdon

Does the method support the Requirements Definition Activity?

Yourdon covers a similar area of the lifecycle to SSADM. There are diagrammatic notations for expressing the requirements of the system based on the three views of data, dynamics and process. The essential model built up is the requirements specification for the system, and this is passed on to the design stages.

There is little support for requirements acquisition, and, as for SSADM, little support for the analysis of requirements. Some analysis is carried out by checking the consistency of the diagrams against the rules of syntax for those diagrams.

What are the main techniques used by the method in support of this activity?

The diagrammatic techniques of the method used to express requirements are :

Data flow diagrams, including the context diagram, which model the processes that transform data in a system and the interfaces between those processes. This is done by emphasising data flowing, being stored and being transformed. This is the process view.

Entity relationship diagrams, which show the relationships between data. This is the data view.

State transition diagrams, which model the dynamic behaviour of the system using the states, transitions, transition conditions and transition actions. This is the dynamics view.

These three views of the system are used to check consistency between them, and guidelines are given on how to do this. Other techniques are :

Process specifications, which specify the processes on the lowest level data flow diagrams.

External event list - which lists those things that occur in the environment to which the system must respond in a predetermined manner. This allows the analyst and the user to identify, specify and agree on relevant events in the systems environment.

Data dictionary - in which precise definitions for all data are given. This contains the meaning, composition and important characteristics of flows and stores on data flow diagrams, and also entities and relationships on entity relationship diagrams.

Is there any existing basis in mathematics for any of the techniques? If not, is there a perceived potential for formalisation?

Data flow diagrams, including the context diagram, in Yourdon are very similar to those of SSADM. While there is no existing basis in mathematics for these, it is believed possible to translate them into Z (see section 4.1 above).

Entity relationship diagrams show relationships between data items, and, as such, may be directly translated into Z. The use of the data composition symbols to describe further the type of relationship involved would help to generate predicates to describe the relationship in Z.

The state transition diagram is the only technique of Yourdon with a mathematical basis. It defines a finite state automaton with output. It is possible to translate this into Z as the technique involves the state, the transition condition, and the transition action, and Z is a state-based approach.

There is no set method for creating process specifications in Yourdon. However, several suggestions are given. There is no reason why such specifications should not be in a formal notation. Rather than use Z for all levels of the data flow diagrams, it may be preferable to specify the lowest level using Z, and then specify the communication between the processes using CSP.

The external event list allows the analyst and the user to identify, specify and agree on relevant events in the systems environment. As such, it may benefit from being written in a notation which has a formal basis, as this will help eliminate ambiguities and misunderstandings. Such a notation does not have to be itself mathematical, as this may not be readily understandable to the user. A similar idea may be useful for the data dictionary, as it is meant to contain precise definitions, and should avoid ambiguities.

Have there been attempts to use the method in conjunction with a formal notation?

It has been suggested [Goldsmith] that the Yourdon structured method and the Vienna Development Method (VDM) could be used together during the system lifecycle. The proposed method of development involves initial development using Yourdon techniques and producing Yourdon models, whilst in parallel supporting these models with VDM specifications to ensure model consistency and correctness. Once an appropriate level of design has been reached, the development towards implementation and code may be carried out purely using VDM refinement (that is, successively making the abstract model more physical). The incorporation of the Yourdon state transition diagram into VDM as a mechanism for specification of control may also be considered.

The expected advantages are: graphical models of system development which are easily understood; mathematical models of system development which may be proved; a structured approach to initial analysis and subsequent design; and consideration of all aspects of system behaviour, including control.

4.4 COntrolled Requirements Expression (CORE)

Does the method support the Requirements Definition Activity?

CORE is aimed at the requirements definition phase. It has been specifically designed to help with the acquisition of the requirements by giving structure to this process, and is the only one of the methods studied to do so. The diagrammatic notations are used to express the requirements, and direct guidelines on which diagrams comprise the final specification is given.

There is some support for requirements analysis. This is achieved by the use of tabular collection forms and the single and combined viewpoint modelling stages. The first of these helps to check for the completeness of the

requirements by ensuring all data has a source and destination viewpoint. It also helps to keep track of the requirements and check for consistency between viewpoints. Both viewpoint modelling stages also help to check for consistency. The combined viewpoint models may also be used to analyse performance characteristics during the constraints analysis stage.

What are the main techniques used by the method in support of this activity?

The diagrammatic techniques used by CORE are :

Viewpoint Structuring, in which a framework for considering the requirements of the target system from the viewpoint of the other systems with which it interacts is provided. This is achieved by creating a viewpoint structure, which is a partitioning of the environment of the target system.

Tabular Collection, in which information on data flow and data transformation is gathered and analysed and the data dictionary created. Using the tabular collections for each viewpoint the analyst can establish completeness of a viewpoint and consistency between viewpoints.

Data Structuring, in which the data is analysed further. This uses a notation derived from Jackson.

Single Viewpoint Modelling, in which the information gathered on the tabular collection forms is transferred to a diagrammatic representation, with new information added about internal data flows, the control of actions and time ordering of actions.

Combined Viewpoint Modelling, in which sequences of operations, or transactions, involving several viewpoints are analysed. Not all possible transactions are considered, but those which involve critical performance or reliability aspects of the proposed system should be analysed.

Constraints Analysis - in which different classes of constraint, such as performance, reliability, etc. are analysed.

Is there any existing basis in mathematics for any of the techniques? If not, is there a perceived potential for formalisation?

None of the techniques of CORE have a sound basis in mathematics. However, there is some potential for formalisation, except for the viewpoint structures.

The tabular collection form describes the input data for a viewpoint, along with the action performed on that data to generate outputs. As such, it is a type of data flow diagram, and may be translated into a formal specification as discussed in sections 4.1 and 4.3 above. Both the single and combined viewpoint model diagrams are also data flow diagrams, and may be treated similarly.

The data structure diagrams use a notation similar to that of Jackson structure diagrams, and it may therefore be possible to translate these into CSP as described in section 4.2 above.

The constraints analysis stage is not very well defined in CORE. The area of non-functional requirements, which is at the heart of this stage, is not very well addressed by any method, formal or informal. Further work needs to be undertaken in this area before any attempt at formalisation can be made.

Have there been attempts to use the method in conjunction with a formal notation?

A comparison of CORE and Z for the specification of requirements has beer carried out by Rachel Edge [Edge] at the Programming Research group at Oxford University for an MSc dissertation. The aim of this project was to investigate requirements analysis by using CORE and Z to specify the same problem. These specifications were then compared and contrasted.

The problem chosen was a hospital monitoring system for which an English specification was given. The dissertation concludes that the Z and CORE specifications cannot be fairly compared because the CORE

specification was done first - this meant that the problem was well understood and a solution formulated before the Z specification was started. Also, the comparison is not like with like, Z and CORE are dissimilar in that CORE combines a notation with a method whereas Z simply provides a notation.

It is worth noting that parts of the formal specification actually used the CSP notation to show how the operations could be sequenced. This use of CSP is not standard practice in Z. It was done because it made the specification easier and more straightforward.

The dissertation concludes with the following suggestions for further work :

a) To see whether the essentials of the CORE method could be separated from the CORE notation and used with a formal notation such as Z.

b) To investigate and further develop the specifications into implementations for comparison.

c) To have the problem re-specified in Z by someone else to see what a specification produced without the aid of the CORE method looks like.

d) To investigate further the way CORE and Z were used.

e) To investigate the use of CSP in the Z specification.

f) To analyse the CORE notations with the aim of defining them better and giving them some semantics.

British Aerospace at Warton are also interested in the comparison between Z and CORE [Bradley]. The report concludes that the use of mathematical methods should help to increase precision and clarity and reduce ambiguity in the specification.

4.5 Soft Systems Approach

Does the method support the Requirements Definition Activity?

The Soft Systems Approach is a methodology aimed at the understanding and analysis of the application domain. It is a way of structuring 'soft' domains using the concept of a human activity system, and of bringing about discussion and comparison of differing views of the world. Conceptual models are built and compared with the real world to suggest feasible and desirable changes.

As such, the Soft Systems methodology supports the requirements acquisition stage of the requirements definition activity.

What are the main techniques used by the method in support of this activity?

The main techniques are :

Expressing the problem situation to build up the richest possible picture of the situation being studied.

Proposing root definitions of relevant systems to suggest possible outlooks on the problem situation, and propose systems which lead to illumination of the problems and hence to their solution or alleviation.

Conceptual modelling in which a model of the activity system needed to achieve the root definition is built.

Comparing the conceptual models with reality to generate a debate about possible changes which might be introduced to alleviate the problem situation.

Is there any existing basis in mathematics for any of the techniques? If not, is there a perceived potential for formalisation?

There is no basis in mathematics for any of the techniques. The only possibility for formalisation is in using a notation to express the problem which had itself been formally defined. This would help to eliminate ambiguities and inconsistencies.

Have there been attempts to use the method *in conjunction with a formal notation*?

There has been no attempt to use the methodology in conjunction with a formal notation. However, techniques based on the soft systems approach have been used at the Admiralty Research Establishment at Portland [Dowle] to derive CORE viewpoint structures for a submarine combat system.

5 Conclusions

During this study of the five selected structured methods, several things have become apparent. The four main methods SSADM, JSD, Yourdon and CORE are similar in many respects. Some of the techniques mentioned in the previous section are used by more than one method. For example, entity life histories are used by JSD and SSADM. Data flow diagrams, in various forms, are used by SSADM, Yourdon and CORE.

There is already work going on at the Programming Research Group at Oxford University to use CSP to give a theoretical basis to JSD, as mentioned in section 4.2 above. This will enable JSD diagrams to be translated into CSP, and vice versa. This translation process could also be applied within SSADM and CORE.

There is little work going on to find a mathematical basis for data flow diagrams, such as those used in SSADM and Yourdon. There is a potential for a translation process into Z, as discussed in section 4.1 above. This would have two aspects:

a) To define a set of rules to convert a data flow diagram into an outline Z specification. The advantage of this approach is that the structure defined by the diagrams will pass over into a well-structured formal specification.

b) To define rules for translating a Z specification into diagrams. The advantage of this approach is that the diagrammatic form may be more readily understood.

However, as Rachel Edge [Edge] discovered, it may be that some parts of the data flow diagrams are best specified using CSP. In this case, a hybrid formal notation incorporating both Z and CSP would need to be used. The development of such a notation would be of great benefit provided the semantics of such a notation could be pinned down.

Also, CORE is the only method, apart from the Soft Systems approach, which offers any support for the requirements acquisition process. It is basically a data flow method, and, as such, may not always be appropriate. It is also the only method which attempts to address the analysis of requirements, albeit in a limited manner. The advantage of a formal notation over the structured methods is that, being written in mathematics, it should be possible to carry out a more detailed analysis of the specification. This would help in checking for completeness, consistency, and the implications and consequences of requirements. Techniques of analysis need to be developed, which may require tool support.

The use of CORE to understand a problem and formulate a solution from which a Z specification can be written has its advantages. However, the suggestion [Edge] that the notations of CORE could be separated from the method and replaced with a formal notation such as Z is much more promising.

There are many techniques, a lot of which have the potential for translating into a formal specification. Unfortunately, this is not the whole story. There must be some way of combining in a well-structured manner the results of such translations into a requirements specification. Without this we will still not have a real "formal

method", just a collection of part-specifications.

6 Further Work

During the course of this study, it has become apparent that further work is needed in all areas of the requirements definition activity.

Requirements Acquisition

There is insufficient support for the requirements acquisition process. The use of the Soft Systems Approach is promising, and should be further investigated.

The process of animating a formal specification may also have a part to play. This could aid in the process of requirements acquisition by demonstrating to the user the implications of his requirements and asking the question "is this what you meant to happen?" It may also aid in the process of requirements analysis. An attempt to animate a specification may lead to failure due to inconsistencies which could then be discovered and resolved. It may also lead to unforeseen effects, for which the causes could be traced and examined.

Requirements Expression

As Rachel Edge [Edge] suggests, further work needs to be carried out in creating a notation using aspects of both Z and CSP. A refinement process which takes this specification towards an implementation would also be useful.

The translation of data flow diagrams into either Z or a Z/CSP hybrid should be further researched and rules derived.

Non-functional requirements are also a part of a requirements specification, and further research is needed to decide the best way of expressing these. This may involve investigating the different sorts of temporal and modal logics available.

Requirements Analysis

There is insufficient support for the requirements analysis process. Techniques for analysis need to be developed, which will almost certainly require tool support. In the case of formal specifications, this may involve theorem-provers.

Production of the Requirements Specification

A method for combining all the formal part-specifications derived from the different techniques of the structured methods needs to be devised.

References

[Bates]
The system engineering layered model - A S Bates (RSRE internal communication)

[Bradley]
"Formal Specification Methods and CORE" - A Bradley. BAe Report number BAe-WSD-GEN-SWE-1268 (1987)

[Bryant]
"Structured and Formal Methods Research Project" - A Bryant and P McGrath. Department of Computing Science, Leeds Polytechnic (1988)

[C3ISSC]
C3I Scientific Support Committee (CERN) Systems Engineering - Methods and Tools Working Group Report on Systems Engineering Issues - Issue 1.0 (April 1989)

[Dowle]
"Derivation of Viewpoint Structure for a Submarine Combat System" - Lt Cdr S W Dowle. ARE TM (UDP) 87102 (1987)

[Edge]
"Comparison of CORE and Z for the Specification of Requirements" - R Edge. MSc Dissertation, Programming Research Group, Oxford University (1986)

[Goldsmith]
"Using the Yourdon Structured Method (YSM) and Vienna Development Method (VDM) Together During the System Lifecycle" - S Goldsmith. Presented at the IEE Colloquium on "The Application of Computer Aided Software Engineering Tools" (1989)

[Hayes]
"Specification Case Studies" - I Hayes (editor). Prentice-Hall International Series in Computing Science (1987)

[Hoare]
"Communicating Sequential Processes" - C A R Hoare. Prentice-Hall International Series in Computing Science (1985)

[Jackson]
"System Development" - M Jackson. Prentice-Hall International Series in Computing Science (1983)

[Jones]
"Systematic Software Development Using VDM" - C B Jones. Prentice-Hall International Series in Computing Science (1985)

[Randell]
"An Introduction to Selected Structured Methods" - G P Randell, RSRE Memo No. 4294 (June 1989)

[Spivey]
"The Z Notation : A Reference Manual" - J M Spivey. Prentice-Hall International Series in Computing Science (1988)

[Sridhar]
"JSD Expressed in CSP" - K T Sridhar and C A R Hoare. Technical Monograph PRG-51, Programming Research Group, Oxford University (1985)

[Strutt]
"A Survey of Formal Methods (for the Specification of Surface Naval Command Systems)" - N Strutt. ARE TM (AXC) 89001 (1989)

DOCUMENT CONTROL SHEET

Overall security classification of sheet .....UNCLASSIFIED.....

(As far as possible this sheet should contain only unclassified information. If it is necessary to enter classified information, the box concerned must be marked to indicate the classification eg (R) (C) or (S))

1. DRIC Reference (if known)	2. Originator's Reference	3. Agency Reference	4. Report Security Classification
	Memo 4293		Unclassified

5. Originator's Code (if known)	6. Originator (Corporate Author) Name and Location
7784000	ROYAL SIGNALS & RADAR ESTABLISHMENT ST ANDREWS ROAD, GREAT MALVERN, WORCESTERSHIRE WR14 3PS

5a. Sponsoring Agency's Code (if known)	6a. Sponsoring Agency (Contract Authority) Name and Location

7. Title

The integration of structural and formal methods

7a. Title in Foreign Language (in the case of translations)

7b. Presented at (for conference papers) Title, place and date of conference

8. Author 1 Surname, initials	9(a) Author 2	9(b) Authors 3,4...	10. Date	pp. ref.
Randell G P			6.1989	16

11. Contract Number	12. Period	13. Project	14. Other Reference

15. Distribution statement

UNLIMITED

Descriptors (or keywords)

continue on separate piece of paper

Abstract

A growing emphasis is being put on a formal, in the sense of mathematically rigorous, approach to the development of software-based systems. However, current formal "methods" tend to be just notations. This memorandum looks at several structured methods and considers the possibilities each offers for integration with a formal notation to produce a usable formal method combining the best of both formal and structured approaches. Proposals for further work are discussed.